Modern Engineering

Day 7

Prudential Financial

GENERAL ASSEMBLY

Whilst we wait....

In chat write:

Your name - role and a recent impulse buy.

LESSON ROADMAP

React State

Making API Requests

Weather API Lab

Styling in React

Material Design

Material Design Lab

MEF MODULE 3 DAY 7: State and Styling in React

Schedule	
9:00–9:15 am	Welcome and Warm-Up
9:15–10:15 am	State Management in React, Mood Points
10:15–10:45 am	**Blog and Posts State**
10:45 am - 12:30 pm	**To-Do List Props and State**
12:30–1:30 pm	Lunch
1:30-1:45 pm	**Making API Calls/Weather Report**
1:45–2:45 pm	**Weather Bonus Features**
2:45 - 3:45 pm	**Material UI Walkthrough**
3:45 - 4:50 pm	**Material UI Lab**
4:50–5:00 pm	Bring It Home

LEARNING OBJECTIVES

1. Manage application state using React's **State hook**
2. Make an API request from a react application
3. Apply **styling** strategies to react components
4. Leverage the **Material Design** UI library in a react application

State and Styling in React

State Management in React

State

- In React, state is an object that holds data that can change within a component.
- State **allows a component to track changes and respond to user interactions** and other events.
- The state can be updated by calling a setter function, which will trigger a re-render of the component and its children.

```
import React, { useState } from "react";

function ExampleComponent() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}

export default ExampleComponent;
```

Image 15: Introduction to Hooks using useState

State vs. Props

State	VS.	Props
Initialized within a component		Passed down from parent
Mutable, but only with special methods		Immutable
Changes trigger re-render		Cannot trigger a re-render

The State-UI Cycle

1. The user is shown an initial page with data and forms/controls
2. Event listeners are triggered when the user activates the given controls
3. The event listeners manipulate the application's state
4. React notices state has changed, and re-renders the relevant components

Getters and Setters with useState

In a react component, state is generated using a special type of function called a **hook**, which we'll discuss next.

The result of calling the **`useState`** function gives you two values:

- A variable allowing **read** access to the value stored in state
- A **setter** function that allows you to reset the value to a new input

```
const [username, setUsername] = useState("")
```


Hooks

- **Hooks** are a way to **add state and other React features to functional components** in React.
- Hooks allow functional components to be used like class components.
- Before hooks were introduced to react v16.8 in 2019, state and functionality was only possible in class components.

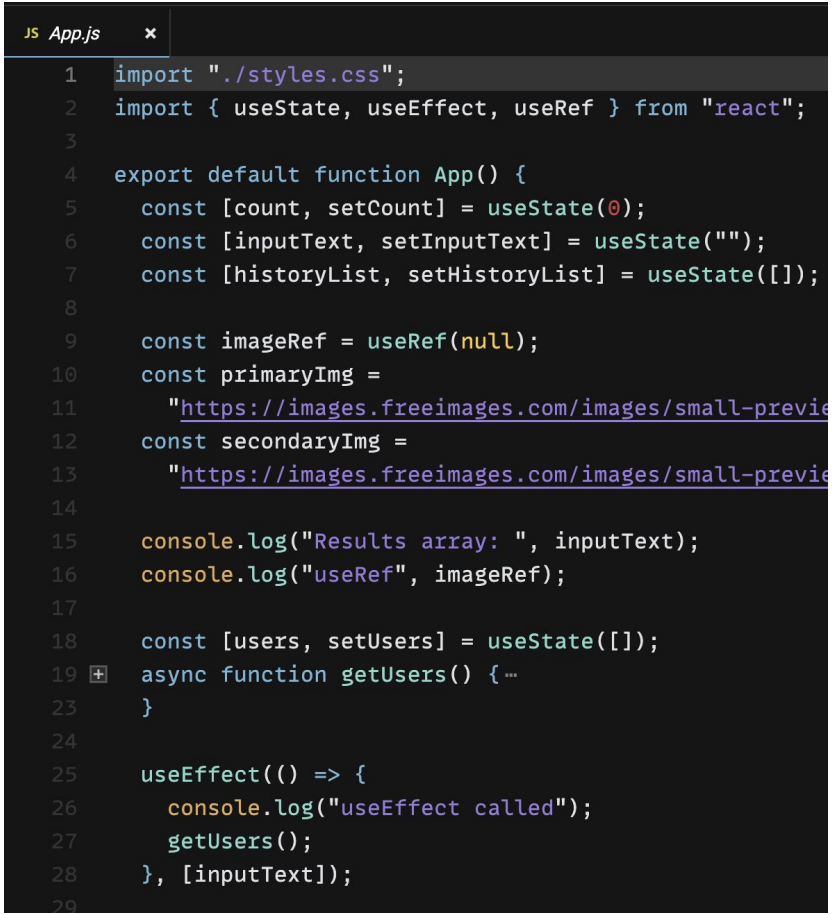


Image: Code block of Hooks functional component

Effect

- Mechanism for triggering a piece of code to run after each *render*.
- **Effects are a way to handle side effects** in React functional components.

```
const [users, setUsers] = useState([]);
async function getUsers() {
  const response = await fetch("https://jsonpl
  const users = await response.json();
  setUsers(users);
}

useEffect(() => {
  console.log("useEffect called");
  getUsers();
}, [inputText]);
```

Image 16: Effect Functional Component

Commonly Used Hooks

Hook	Definition
useState():	Used to **add** state to a functional component. It takes an initial value as an argument, and returns an array containing the current state and a function to update it.
useEffect():	Used to **run side effects** such as **fetching data**, updating the document title, or subscribing to a message service. It takes a function that contains the logic for the effect and an array of dependencies as arguments.
useContext():	Used to **consume data** from a context object. It takes the context object as an argument and returns the current value of the context.

Guided Walk-Through:

Mood Points Increasing

Let's practice initializing and updating state in response to user actions

Partner Exercise:

Blog State and Forms

30 minutes

Practice using a state-based value and updating it in response to user actions

Guided Walk-Through:

To Do List: Props and State

Let's solidify these foundational concepts in the 03 and 04 READMEs by creating the start of a To Do List application

Partner Exercise:

React Hooks Continued (Optional time permitting)

30 minutes

[Here is a self-guided guide to additional React Hooks]

Lunch

State and Styling in React

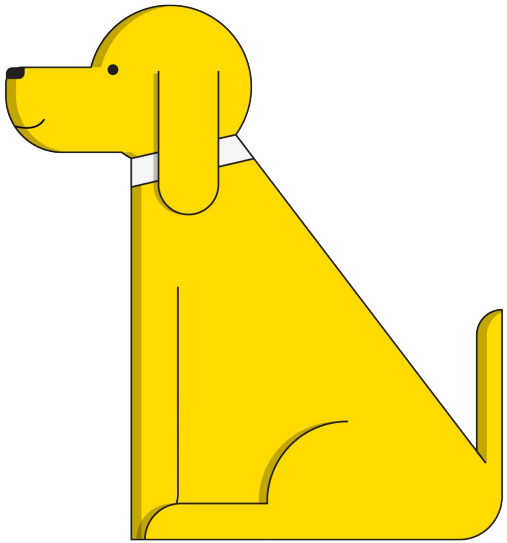
Making API Calls from React

Fetching Data With `fetch()`

From react or any front-end webpage, you can send an HTTP request using JavaScript's built-in **`fetch()`** function, which accepts the destination URL as a parameter and sends a GET request to it:

```
const response = await fetch(url);
```

You can also provide a second parameter, an options object, for requests with different HTTP methods or requiring special headers.



Asynchronous Code With Async/Await

API calls are **asynchronous** code. Because we have no idea how long the response will take to return, we have to **await** the result. We can do so using two keywords that come as a pair: `async` and `await`.

```
async function askForData(){
    const response = await fetch(url);
    const data = await response.json();
}
```


Guided Walk-Through:
Weather Report

We'll create a searchable weather application that relies on the WeatherStack API to fetch and display weather information to the user based on their search input.

Partner Exercise:

Extending the Weather Report

60 minutes

[Let's add bonus features to the Weather App] to continue practicing the use of forms and API calls in React applications

Partner Exercise:

APIs + Fetch - Choose Your Own Adventure

60 minutes

Continue practicing the use of forms and API calls in React applications. Feel free to create a new app or continue adding on to the Weather App we created.

- Chuck Norris Joke API Exercise
- Dog CEO API Exercise
- Giphy API Exercise
- Weather App Additional Features

Recap

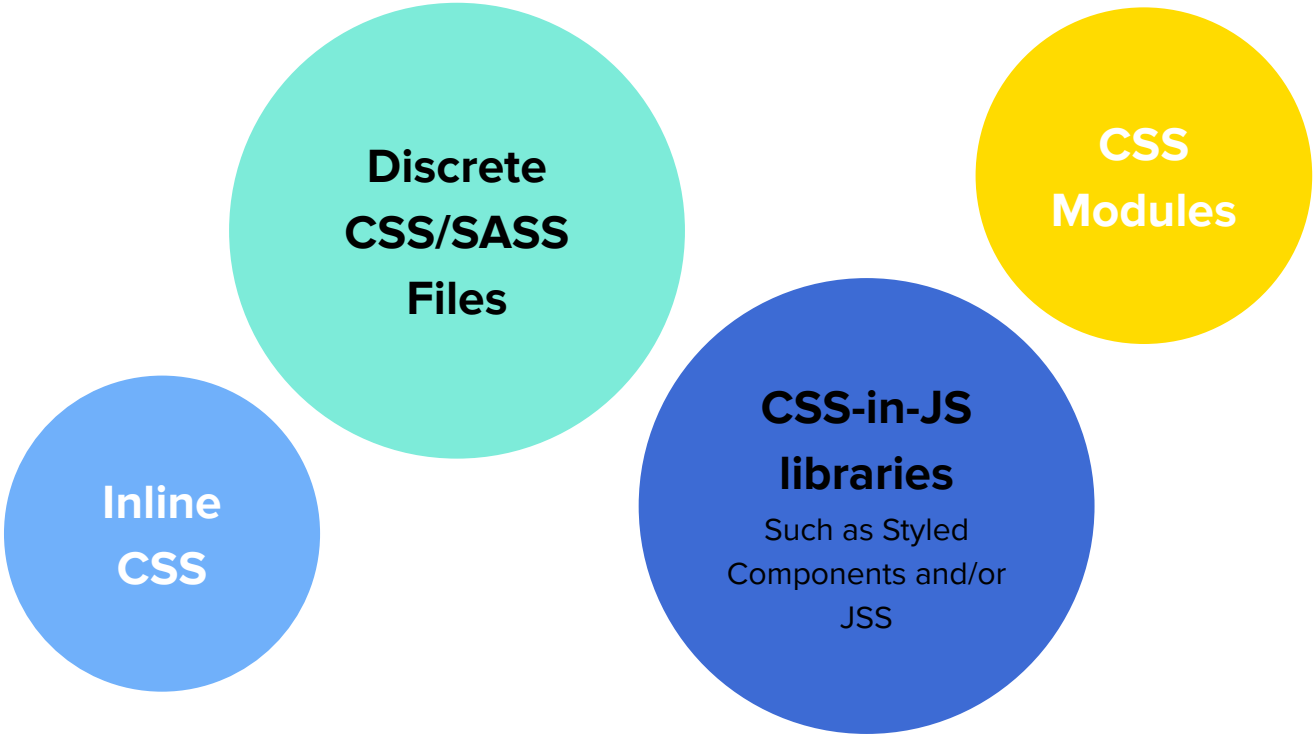
- State vs. Props
- useState
- fetch

Any Questions

Modern Engineering

Styling Approaches in React

Styling Approaches



Styling Approaches: Pros and Cons

Approach	Pros	Cons
Inline CSS	• Direct control over a specific HTML element. • Quick and easy to apply to a single element.	• Increases HTML code size and complexity. • Makes it difficult to maintain a consistent look and feel across multiple pages. • Lack of reusability, as the same styles cannot be applied to multiple elements or pages. • Styles cannot be overridden easily.

Styling Approaches: Pros and Cons (con't)

Approach	Pros	Cons
Discrete CSS/SASS Files	• Reusability of styles across multiple pages or elements. • Easy to maintain and update a consistent look and feel across the site. • Better organization of styles. • Improved performance due to caching. • Ability to use advanced features like variables, functions, and mixins.	• Initial setup and setup of a build system may be complex. • Requires a separate HTTP request for each CSS file, increasing page load time. • Can make it difficult to quickly apply styles to a specific HTML element. • Requires coordination between HTML, CSS, and JavaScript files.

Styling Approaches: Pros and Cons (con't)

Approach	Pros	Cons
CSS-in-JS Libraries	• Scoped styling, reducing the risk of style conflicts. • Dynamic styling based on component state or props. • Improved modularity and encapsulation. • Access to JavaScript functions and variables in CSS. • Easy management of media queries and other complex styling.	• Increased bundle size due to the need to include a library. • Requires a learning curve for developers unfamiliar with the library. • May result in reduced performance due to the overhead of JavaScript styling. • Limited browser compatibility for some libraries. • May make it difficult to collaborate with designers who are unfamiliar with the library.

Styling Approaches: Pros and Cons (con't)

Approach	Pros	Cons
CSS Modules	• Encapsulation of styles, reducing the risk of style conflicts. • Dynamic composition of styles based on component state or props. • Reusability of styles across multiple components. • Improved modularity and organization. • Easy management of media queries and other complex styling.	• Requires a build step to compile CSS modules into CSS. • Limited browser compatibility without a build step. • Increased bundle size due to the need to include a CSS module in the JavaScript code. • Can make it difficult to collaborate with designers who are not familiar with the CSS Modules concept. • Limited browser dev tools support for debugging and inspecting CSS Modules.

Discussion:

Styling Approaches

10 minutes

Of the styling approaches we've outlined, which would be the most appropriate for Prudential?

Come off mute and discuss!

- **Inline CSS**
- **Discrete CSS/SASS Files**
- **CSS-in-JS libraries**
  Such as Styled Components and/or JSS
- **CSS Modules**

Modern Engineering

Material Design

Material Design: Make Your Apps Googlier

Material Design is a UI library that allows you to easily integrate a highly-engineered set of UI components, and their accompanying style, into any project.

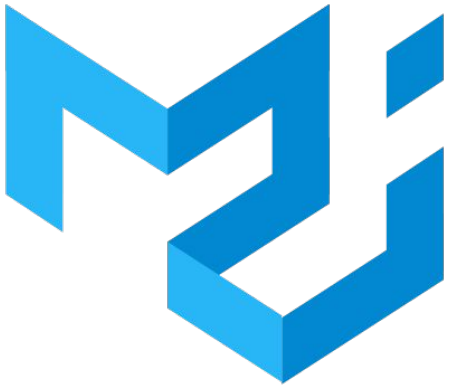
Material Design was created by Google to power the consistently "google" feel across their many applications and websites. It provides easily recognizable interface elements that users are already comfortable with.

Guided Walk-Through:
Add Material UI to a React Application

1. Let's create another React Application we can style using the Material UI Component Library.

2. We will practice reviewing by following this walkthrough

3. Follow along as we code together and remember to ask questions if you get stuck.



Solo Exercise:

Add Material UI to a React Application (Lab)

- Choose any React app you've built in the course that could benefit the most from enhanced styling and user-friendly components.
- Follow along with specific instructions in this repository
- Your goal will be to:
  - Plan the Integration
  - Integrate Material UI
  - Test and review your work
- You may work **independently** or **in group** depending or your preference.

If you get stuck or need some help, we are available to provide assistance and troubleshoot problems.

Recap

- Adding Material UI to a React project
- Making API calls with fetch
- State and Hooks

Any Questions


GA